Making REST APIs

Your First REST API with Laravel

API Route = URL that returns JSON (Recap)

Example:

```
http://localhost:8000/api/hello
```

Returns:

```
{
    "message": "Hello Laravel API!"
}
```

That's it! No HTML, just data.

What You Write vs What You Call:

File: `routes/api.php`	Actual URL
`Route::get('/hello')`	`http://localhost:8000/api/hello`
`Route::get('/test')`	`http://localhost:8000/api/test`
`Route::get('/students')`	`http://localhost:8000/api/students`

API Route and Controller

Instead of processing the endpoints, Laravel invokes the controller method.

```
  Route::get('/students', [StudentController::class, 'index']);
```

Returns a JSON response with student data

Secure API Route

API Setup with Laravel Sanctum

Laravel provides `php artisan install:api` to set up API authentication with Sanctum.

```
php artisan install:api --force
```

The `routes/api.php` file is generated to manage users who can access the secure routes.

```php
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

Route::get('/user', function (Request $request) {
    return $request->user();
})->middleware('auth:sanctum');
```

- Enables API routes in bootstrap/app.php automatically (we did it manually before)

```php
<?php

use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;

return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        api: __DIR__.'/../routes/api.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(function (Middleware $middleware): void {
        //
    })
    ->withExceptions(function (Exceptions $exceptions): void {
        //
    })->create();
```

- Generates `create_personal_access_tokens_table` to run the migrations.

```
  INFO  Running migrations.

0001_01_01_000000_create_users_table ........................... 43.55ms DONE
0001_01_01_000001_create_cache_table ........................... 10.83ms DONE
0001_01_01_000002_create_jobs_table ............................ 25.07ms DONE
2025_09_17_021049_create_students_table ......................... 3.19ms DONE
2025_09_17_021112_create_personal_access_tokens_table ......... 15.36ms DONE
```

Additionally, it Installs API authentication (not APIs themselves)

- Installs the Sanctum package
- Adds Sanctum middleware
- Publishes Sanctum configuration
- Adds Sanctum authentication middleware (not general API routes)
- Enables token-based authentication for API routes

Create a User class

We need to update a User class in the app/Models directory

The User class should use the `HasApiTokens`.

```php
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasFactory, HasApiTokens, Notifiable; // <-
    protected $fillable = ['name', 'email', 'password'];
    protected $hidden = ['password', 'remember_token'];
    protected function casts(): array {
        return ['email_verified_at' => 'datetime', 'password' => 'hashed'];
    }
}
```

Create Secure API Routes

Add routes to `routes/api.php`:

```php
<?php
use Illuminate\Support\Facades\Route;

// Protected endpoint (requires bearer token)
Route::get('/goodbye', function () {
    return response()->json([
        'message' => 'Goodbye Laravel!',
        'user' => auth()->user()->name
    ]);
})->middleware('auth:sanctum');
```

Notice that adding only one line `->middleware('auth:sanctum');` enables the Bearer token API processing.

You can make the goodbye() method in the StudentController and call it from the route.

```
    public function goodbye()
    {
      return response()->json([
        'message' => 'Goodbye Laravel!',
        'user' => auth()->user()->name
      ]);
    }
```

```
Route::get('/goodbye', [StudentController::class, 'goodbye'])
->middleware('auth:sanctum');
```

Generate Bearer Token

Run the `generate-token.sh` script to generate the bearer token for the Test user.

```php
<?php
require_once 'vendor/autoload.php';
$app = require_once 'bootstrap/app.php';
$app->make('Illuminate\Contracts\Console\Kernel')->bootstrap();

use App\Models\User;

$user = User::factory()->create([
    'name' => 'Test User',
    'email' => 'test@example.com'
]);

$token = $user->createToken('api-token')->plainTextToken;
echo $token;
```

Why Use Bearer Tokens?

- **Stateless by design** → APIs don’t rely on server-side sessions.
- **Cross-platform support** → works seamlessly across mobile, web, and external services.
- **No cookie/session dependency** → avoids browser-specific limitations.
- **Scalable** → easier to handle many users without managing sessions.
- **SPA friendly** → ideal for single-page applications where cookies are less convenient.
- **Machine-to-machine ready** → simplifies authentication between services.
- **Microservices fit** → lightweight, portable, and designed for distributed systems.

Test Protected Endpoint (With Bearer Token)

```
curl -H "Authorization: Bearer YOUR_TOKEN_HERE" \
     http://localhost:8000/api/goodbye
```

Response:

```
{
    "message": "Goodbye Laravel!",
    "user": "Test User"
}
```

Status Code: 200 OK

Test Protected Endpoint (Without Token)

```
curl http://localhost:8000/api/goodbye
```

Response:

```
{
    "message": "Unauthenticated."
}
```

Status Code: 401 Unauthorized

Summary of Using Protected/Secure Endpoint

1. Run `php artisan install:api`
2. Add HasApiTokens to User model
3. Create API route with auth:sanctum middleware,
4. Generate token using createToken()
5. Use the API with Authorization: Bearer header

Analyzing run1-6.sh

⚠️ WSL2 Warning: Run `dos2unix` command before running any scripts.

Copy and Run the Script

To run this script, copy the corresponding directory and script to the directory that you test.

```
# in the temp/ase230 directory (for example)
bash run1-6.sh # run script
cd student-api
php artisan serve

#WSL2:
php artisan serve --host=0.0.0.0
```

Open another shell or terminal

In the project directory:

```
> curl -H "Authorization: Bearer $(cat api_token.txt)" http://localhost:8000/api/goodbye
{"message":"Goodbye Laravel!","user":"Test User"}

> php artisan route:list --path=api

  GET|HEAD  api/goodbye ..................................................
  POST      api/greet ............................ Api\StudentController@greet
  GET|HEAD  api/hello ............................ Api\StudentController@hello
  GET|HEAD  api/students ......................... Api\StudentController@index
  POST      api/students ......................... Api\StudentController@store
  GET|HEAD  api/students/major/{major} ...... Api\StudentController@getByMajor
  GET|HEAD  api/students/search ................. Api\StudentController@search
  GET|HEAD  api/students/stats .................. Api\StudentController@stats
  GET|HEAD  api/students/year/{year} ......... Api\StudentController@getByYear
  GET|HEAD  api/students/{student} ............... Api\StudentController@show
  PUT       api/students/{student} .............. Api\StudentController@update
  DELETE    api/students/{student} ............. Api\StudentController@destroy
  GET|HEAD  api/test .....................................................
  GET|HEAD  api/time ............................ Api\StudentController@time
```

Analysis of the Script

Variables Setup

```
SOURCE_DIR="student-api1-6"
TARGET_DIR="student-api"
MYSQL_PASSWORD=123456
DB_NAME="laravel_app"
DB_USER="laravel_user"
DB_PASSWORD="password123"

if [[ "$OSTYPE" == "darwin"* ]]; then
  MYSQL="mysql"
else
  MYSQL="sudo mysql"
fi
```

Set the error flag.

```
set -e # Exit immediately if any command fails
```

Get password if the password is not in the variable

```
if [ -z "$MYSQL_PASSWORD" ]; then
    echo "Creating database..."
    read -s -p "Enter MySQL root password: " MYSQL_PASSWORD
    echo
fi
```

Create database and user

1. Drop (remove) DB if the laravel_app already exists to start afresh.
2. Create DB laravel_app.
3. Create the user laravel_user for the DB.

```
# Drop and recreate the database completely to ensure a clean state
$MYSQL -u root -p$MYSQL_PASSWORD << EOF
DROP DATABASE IF EXISTS $DB_NAME;
CREATE DATABASE $DB_NAME;
CREATE USER IF NOT EXISTS '$DB_USER'@'localhost' IDENTIFIED BY '$DB_PASSWORD';
GRANT ALL PRIVILEGES ON $DB_NAME.* TO '$DB_USER'@'localhost';
FLUSH PRIVILEGES;
EOF
```

Creating a Laravel project

```
composer create-project laravel/laravel $TARGET_DIR
```

Change directory to the generated Laravel project and update the .env file

```
cd $TARGET_DIR

sed -i 's/DB_CONNECTION=sqlite/DB_CONNECTION=mysql/' .env
sed -i "s/# DB_DATABASE=laravel/DB_DATABASE=$DB_NAME/" .env
sed -i "s/# DB_PASSWORD=/DB_PASSWORD=$DB_PASSWORD/" .env
sed -i 's/# DB_HOST=127.0.0.1/DB_HOST=127.0.0.1/' .env
sed -i 's/# DB_PORT=3306/DB_PORT=3306/' .env
sed -i "s/# DB_USERNAME=root/DB_USERNAME=$DB_USER/" .env
sed -i 's/SESSION_DRIVER=database/SESSION_DRIVER=file/' .env
sed -i 's/CACHE_DRIVER=database/CACHE_DRIVER=file/' .env
```

Now, we have the correct .env DB section:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel_app
DB_USERNAME=laravel_user
DB_PASSWORD=password123
```

Create app/Http/Controllers/Api Directory

```
CONTROLLER_DIR=app/Http/Controllers
mkdir -p $CONTROLLER_DIR/Api
```

Generate Student Controllers for API

```
php artisan make:controller StudentController --api
php artisan make:controller api/StudentController --api
```

Generate Student Model without Migration

We don't create migration (-m) this time.

```
php artisan make:model Student # No migration file this time (-m)
```

Generate the bootstrap/app.php

```
php artisan install:api --force --no-interaction
```

Copy files from existing code

Instead of writing code from scratch, we copy the code from existing directory.

```
echo "Copying Controllers..."
SRC_PATH=$CONTROLLER_DIR/Api/StudentController.php
SRC_PATH=$CONTROLLER_DIR/StudentController.php
cp "../$SOURCE_DIR/$SRC_PATH" $SRC_PATH

echo "Copying Models..."
SRC_PATH=app/Models/Student.php
cp "../$SOURCE_DIR/$SRC_PATH" $SRC_PATH
SRC_PATH=app/Models/User.php
cp "../$SOURCE_DIR/$SRC_PATH" $SRC_PATH
```

```
echo "Copying routes/api.php with test routes..."
SRC_PATH=routes/api.php
cp "../$SOURCE_DIR/$SRC_PATH" $SRC_PATH
SRC_PATH=routes/web.php
cp "../$SOURCE_DIR/$SRC_PATH" $SRC_PATH

echo "Copying DB migrations/seeders..."
SRC_PATH=database/seeders/StudentSeeder.php
cp "../$SOURCE_DIR/$SRC_PATH" $SRC_PATH

echo "Copying Generate Token Script..."
SRC_PATH=generate-token.sh
cp "../$SOURCE_DIR/$SRC_PATH" $SRC_PATH
```

Apply migration template

Create the migration for the Student Model.

```
echo "Setting up migration..."
php artisan make:migration create_students_table --create=students --quiet
```

We need to fill in the up() method, but instead, we copy the file from the existing template file.

```
# We can find the migration file in the database/migrations
MIGRATION_FILE=$(find database/migrations -name
  "*_create_students_table.php" | sort | tail -1)
TEMPLATES="../$SOURCE_DIR/database/students_migration_template.php"
cp "$TEMPLATES" "$MIGRATION_FILE"
```

Run migrations to generate DB tables

```
echo "Running migrations..."
php artisan migrate --force --no-interaction
```

Create Seed Student example

```
echo "Running seeders..."
php artisan db:seed --class=StudentSeeder --force --no-interaction 2>/dev/null
```

Clear cache

```
php artisan optimize:clear
```

Verifying API routes

```
php artisan route:list --path=api
```

Run generate-token.sh to generate a token

```
bash "generate-token.sh"
```

Display message

To start the server:

```
cd $TARGET_DIR"
php artisan serve
# for WSL1
php artisan server --host=0.0.0.0
```

To test API endpoints:

```
curl http://localhost:8000/api/test"
curl http://localhost:8000/api/hello"
curl http://localhost:8000/api/hello/YourName"
```

To see all API routes:"

```
php artisan route:list --path=api"
```

To test Bearer Token API endpoints:"

```
curl -H \"Authorization: Bearer \$(cat api_token.txt)\" http://localhost:8000/api/goodbye"
```

Add new REST APIs

Now, let's learn how to create your own custom REST API endpoints step by step.

Step 1: Add Route in `routes/api.php`

Choose the endpoints/methods and corresponding controllers.

```
Route::get('/example', [CourseController::class, 'example']);
```

Step 2: Create Controller in `app/Http/Controllers/Api`

```
class CourseController extends Controller
{
    public function example() { ... }

}
```

Step 3: Update Model (Optional)

This is necessary only when you change the update model; however, in this case, it might be a better option to run the whole script.

1. Update Model in `app/Models`.
2. Update Migration in `database/migrations`.

```
// database/migrations/create_courses_table.php
public function up()
{
    Schema::create('students', function (Blueprint $table) {
      ...
    });
}
```

Step 4: Add Authentication (Optional)

Protect specific routes:

```
// In routes/api.php
Route::post('/example', [CourseController::class, 'example'])
    ->middleware('auth:sanctum'); // Protected
```

Only the bearer token is needed to access the secured API.